

Object-Oriented Design

Object-Oriented Concepts **Part 1 - Object**

1

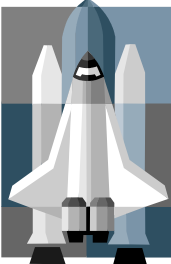
Basic Knowledge

Object-oriented concepts

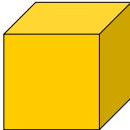
- ✓ objects: state, behaviour, identity
- ✓ relationships between objects
- ✓ classes: inside, outside (public interface)
- ✓ class relationships: inheritance, using, instantiation, meta class
- ✓ (class) attributes, (class) methods – operations

2


Object-oriented concepts : Objects




a rocket



a cube




Peter's wrench



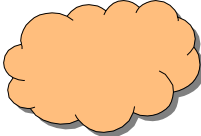
Tricia's wrench

Article	Price
coat	275
dress	325
suit	412

a price table



Anna



a cloud

3

Object-oriented concepts : Objects

- An object complies at least with one of the following criteria:
 - A **tangible** and or **visible** thing
 - something that can be **intellectually understood**
 - something **that can be thought of** or something that **can be operated upon**
- Objects can be simple (a needle, a line), but it can also be complex (a chemical factory, an accounting system , a car) and have complex behaviour
- Some objects are tangible and have clear borders, others can be more fuzzy : a lake, the fog, a crowd .
- **Colour, emotion, time, ... are not objects**, but can be properties of objects, e.g.: "Tricia has blue eyes", "Charles loves water".

4

Object-oriented concepts : Objects

State

- **An object has state.**
The state consists of all (mostly static) properties of an object together with the actual value (mostly dynamic) of all these properties.
An example:
 - An vending machine for coffee at a given moment contains a certain amount of coffee, water, cups, sugar, ... and money. When someone inserts money in the machine, it remembers the inserted amount. When the customer makes a choice for a certain type of coffee and the amount inserted is insufficient, the machine does nothing. ...
 - A static property of the coffee distribution machine is that it accepts money.
 - The amount of money inserted at a given moment is a dynamic value of that property
- **The state of an object has an influence on the behaviour!**

Object-oriented concepts : Objects

Behaviour

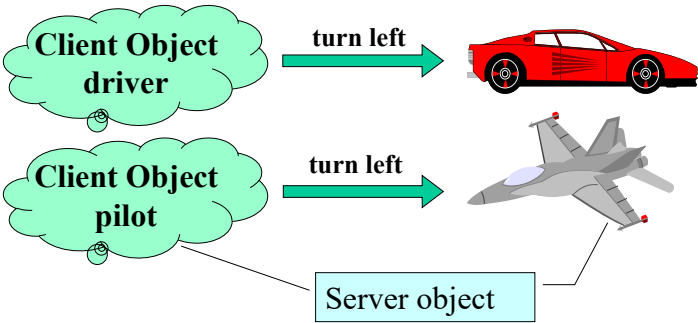
- **An Object has behaviour.** It reacts to questions and asks questions to other objects. The behaviour of an object is completely defined by its actions and reactions, that in turn are influenced by the state of the object.
- A **operation** is an action performed by an object on an other object. Another way of putting this is that an object sends a message to an other object (message passing system). The two expressions are equivalent.
- A method is the implementation of an operation. In Java it is called as such.

6

Object-oriented concepts : Objects

Behaviour

The actual content of an operation (the implementation of the operation or the method) depends on the receiving object. ▶ **Polymorphism**

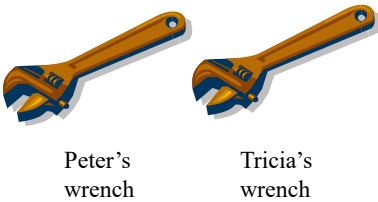


7

Object-oriented concepts : Objects

Identity

Every object has a unique identity.

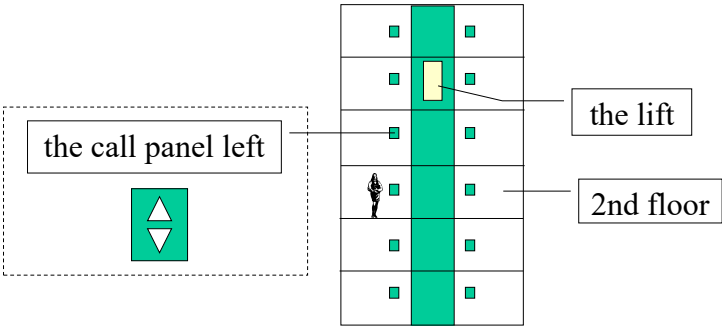


- Even though two objects are "identical",- this is: on the basis of their properties it is impossible to make a difference-, they still remain two different objects.

8

Object-oriented concepts : relationships between objects

- **Objects as such are completely uninteresting.**
Co-operating objects define the global behaviour of the system.
- Objects rely on each other for delivering services (**using relationship**)



9

Object-oriented concepts : relationships between objects

Someone on the second floor wants to go to the ground floor and pushes the descend button on the call panel left from the lift shaft.

The light of the call panel indicating that someone called the lift to go down and the panel on the otherside of the door turns on. The call panel also sends a request to the lift to come to the second floor.

When the lift arrives the door opens ...

10

Object-oriented concepts:
relationships between objects

In order to communicate with one another the different objects must “**know**” each other. A call panel left must know the call panel right and vice versa. Every call panel must know the lift. But the lift doesn’t know the call panels.

an UML object diagram shows the relationship between different objects

The diagram shows three objects: 'the call panel left on the 2nd floor', 'the call panel right on the 2nd floor', and 'the lift'. There is an undirected association between the two call panels. There are directed associations from each call panel to the lift, indicated by arrows and an 'X' at the target end. Callouts explain: 'link has no arrow directions not determined' for the undirected link, and 'A directed link has an arrow and an X' for the directed links. The text 'UML 2.0' is also present.

11

Object-oriented concepts : relationships between objects

- An object can contain other objects (**containing relationship**)
- In the previous example the different parts can also be seen as part of the elevator system:
 - the **motor**
 - the **sensor** that has to know where the elevator is .
 - the **control panel** that accepts requests to go to a certain floor , or to open or close the doors.
 - ...

The diagram shows 'the lift' at the top, connected by a line with an open diamond to three objects below: 'the motor', 'the sensor', and 'the control panel', indicating that the lift contains these components.

12

Object-oriented concepts : relationships between objects

Aggregation

- **Aggregation** is when one class is used as part of another class, but still exists outside of that other class.

13

Object-oriented concepts : relationships between objects

Aggregation

- **an aggregation**
 - If the car moves the motor (being a **part of** the car) also moves...
 - If I sell the car I sell it as a whole
 - Yet if the motor breaks down I can buy a new motor and replace the old one

The parts can exist without the whole



14

Object-oriented concepts : relationships between objects

Composition

- In composition, the object composed of other behaviors owns those behaviors. When the object is destroyed, so are all of its behaviors.

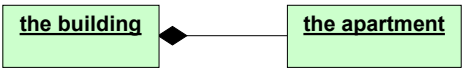
15

Object-oriented concepts : relationships between objects

Composition

- **a composition**
 - The building has many floors and on every floor there are several apartments.
 - I can buy or sell an apartment
 - But if the building is demolished, then the apartments also are.

the parts depend on the whole to merely exist



16

- **Composition or aggregation** allows you to use behavior from a family of other classes, and to change that behavior at runtime.

- **Composition or Aggregation?**

Ask yourself: does the object whose behavior I want to use exist outside of the object that uses its behavior?